项目结构与后端构建指引

一、项目结构与文件构建（参考操作）

1. 可使用 Controller-Service-Mapper 的结构进行构建。
2. Controller 使用@RestController 与@RequestMapping 注解
3. SeviceImpl 使用@Service 注解
4. Mapper 使用@Component 注解
5. 后端返回格式建议统一方便前端操作，可以如下：

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Result {
    private Integer code;//响应码，1 代表成功；0 代表失败
    private String msg;  //响应信息 描述字符串
    private Object data; //返回的数据

    //增删改 成功响应
    0 个用法
    public static Result success() { return new Result( code: 0,  msg: "success",  data: null); }

    //查询 成功响应
    32 个用法
    public static Result success(Object data) { return new Result( code: 0,  msg: "success", data); }

    //失败响应
    public static Result error(String msg) { return new Result( code: -1, msg,  data: null); }
```

6. 在 application.yml 文件中填写以下内容

```
server:
  port: 6253
```

7. application.yml 与 application.properties 文件均放入 Resources 文件夹中。

二、关于调用 idme 的 api 操作方法

iDME 全量数据服务 API 封装文档：

https://codelabs.developer.huaweicloud.com/codelabs/home?pageno=1&keyword=idme

可根据文档中的 Delegator、Feign 和 RestTemplate 三种方法任一选择一种进行开发。

（1）在运行态中点开左侧数据服务管理->全量数据服务

（2）选择希望实现功能的接口，如下。

关注下面的请求、响应参数与请求、响应实例，查看是否有必须传入的参数

请求参数

	名称	类型	参数位置	是否必须	示例	描述
1	params	对象	body	否	-	-
2	id	数值	body	是	0	ID
3	modifier	文本	body	否	"string"	处理人
4	applicationId	文本	body	否	"string"	应用ID

响应参数

	名称	类型	示例	描述
1	result	文本	"SUCCESS"	result
2	data	数组	-	data
3	errors	数组	[]	-

三、Delegator 与 Service 使用方法

1.Delegator 使用方法为在类中注入 Bean 对象

```
@Autowired
private PartDelegator partDelegator;
```

2.在方法中调用 Delegator 函数：

例：partDelegator.create()

3.在文件服务中需要使用以下服务：

```
@Autowired
private FileDelegatorService fileDelegatorService;
```

四、数据模型 DTO 对象

1.数据实例接口实例

（1）创建数据实例

对于数据模型 x，使用 xdelegator.create(xCreateDTO)进行创建操作。其中可根据以下说明灵活使用相应的 DTO 对象。

DTO对象说明

- xxxCreateDTO：创建数据实例时使用的数据传输实体，用于创建接口（create/batchCreate）入参。
- xxxHistoryViewDTO：数据实例历史记录实体数据传输实体，用于系统版本查询接口（queryHistoryData）返回接收对象。
- xxxListViewDTO：基础属性视图数据传输实体，用于基础属性查询接口（list）返回接收对象。
- xxxQueryViewDTO：列表属性视图数据传输实体，用于列表属性查询接口（query）返回接收对象。
- xxxSaveDTO：执行数据实例保存时使用的数据传输实体，用于保存接口（save）入参。
- xxxSaveAllDTO：批量执行数据实例保存时使用的数据传输实体，用于批量保存接口（saveAll）入参。
- xxxSaveAsDTO：另存为数据实例时使用的数据传输实例，用于另存为接口（saveAs）入参。
- xxxUpdateDTO：更新数据实例时使用的数据传输实体，用于更新接口（update）、批量更新接口（batchUpdate）、条件更新接口（updateByCondition）入参。
- xxxViewDTO：实例属性视图传输实体，用于需要全量返回数据实例属性接口的返回接收对象，如创建/更新等。

如：

```
public boolean createProduct(ProductViewDTO product) {
    ProductCreateDTO p = new ProductCreateDTO();
    // 名称
    if (product.getProductName() != null && !product.getProductName().isEmpty()) {
        p.setProductName(product.getProductName());
    }
    // 负责人
    if (product.getProductOwner() != null && !product.getProductOwner().isEmpty()) {
        p.setProductOwner(product.getProductOwner());
    }
    // 过程阶段
    if (product.getProductStage() != null) {
        p.setProductStage(product.getProductStage());
    }
    //产品信息
    if(product.getProductInformation()!=null && !product.getProductInformation().isEmpty())
    {
        p.setProductInformation(product.getProductInformation());
    }

    ProductViewDTO createProduct=productDelegator.create(p);

    return createProduct!=null;
}
```

五、调用 Delegator 和 Service 注意事项

1.更新操作。可根据 id 为索引，updateDTO 可修改输入的参数数据，未输入的参数数据不会修改。

注意：part 的更新需要使用 updateByadmin（update）

2. 获取实例信息。可使用对应类辅助进行查询操作。可灵活操作 condition 和 rdmPageVo 满足需求。

```
QueryRequestVo queryRequestVo = new QueryRequestVo();
queryRequestVo.addCondition( name: "id", ConditionType.EQUAL, id);
RDMPageVO rdmPageVO = new RDMPageVO( curPage: 1, Integer.MAX_VALUE);
List<ProductViewDTO> list = productDelegator.find(queryRequestVo, rdmPageVO);
```

```
QueryRequestVo queryRequestVo = new QueryRequestVo();
queryRequestVo.addCondition( name: "productName", ConditionType.LIKE, name);
RDMPageVO rdmPageVO = new RDMPageVO( curPage: 1, Integer.MAX_VALUE);
List<ProductViewDTO> list = productDelegator.find(queryRequestVo, rdmPageVO);
```

3. 删除操作

可以使用 delegator.deleteByCondition(DeleteByCondition d)执行删除操作。

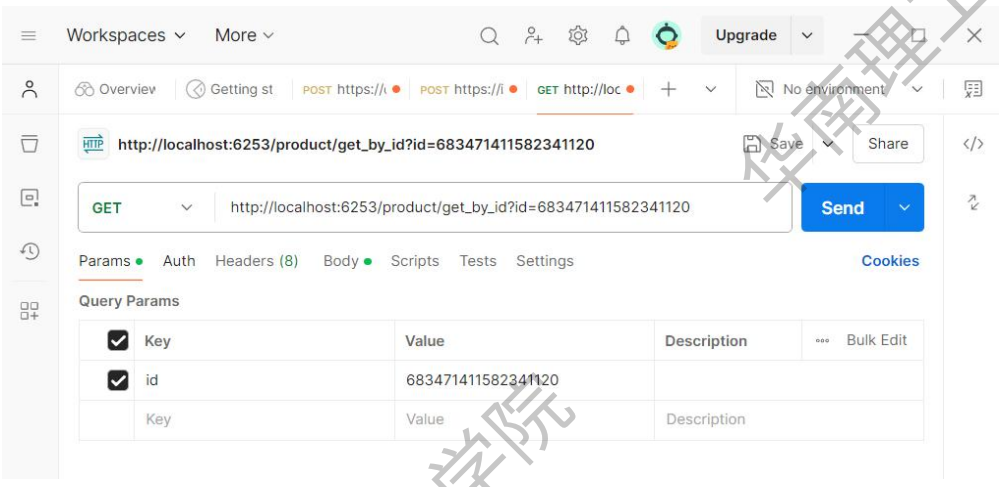
4. 前后端交互接口。根据前端传入参数的不同灵活选择不同的接收方式。

```
@PostMapping("/update")
public Result update_Product(@RequestBody ProductUpdateDTO updateDTO)
@GetMapping("/get_by_id")
public Result get_ProductById(@RequestParam("id") long id)
```

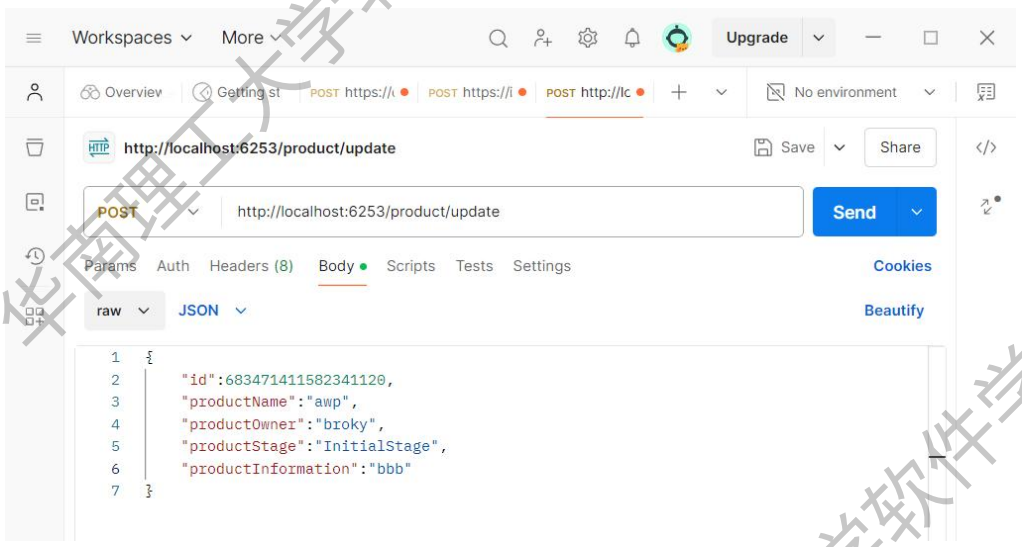
5. 调用接口的方式

（1）创建。传入的 JSON 格式可通过 java 调试获取参数需求或通过下面的查询获取的返回 JSON 格式进行调整。

（2）查询



（3）修改。可根据 id 为索引，updateDTO 可修改输入的参数数据，未输入的参数数据不会修改。



注：更新 part 的 json 格式与查询返回的响应格式不同，应如下图：

6. 部件的分类及其属性操作的实现

（1）存在主对象和分支对象的数据模型 part，需在创建部件时，PartcreateDTO p 需进行下列操作。

```
PartMasterCreateDTO pmcd = new PartMasterCreateDTO();
p.setMaster(pmcd);
PartBranchCreateDTO pbcd = new PartBranchCreateDTO();
p.setBranch(pbcd);
```

（2）获取分类列表

```
public ArrayList<ClassificationNodeViewDTO> getClassificationList() {
    QueryRequestVo queryRequestVo = new QueryRequestVo();
    queryRequestVo.addCondition( name: "businessCode", ConditionType.LIKE, value: "");
    RDMPageVO rdmPageVO = new RDMPageVO( curPage: 1, Integer.MAX_VALUE);
    return (ArrayList<ClassificationNodeViewDTO>) classificationNodeDelegator.find(queryRequestVo, rdmPageVO);
}
```

（3）根据分类 ID 获取其属性。

```
public List<EXADefinitionLinkViewDTO> getAttributeLinkByTargetID(String id) {
    QueryRequestVo queryRequestVo = new QueryRequestVo();
    queryRequestVo.addCondition( name: "target.id", ConditionType.EQUAL, id);
    RDMPageVO rdmPageVO = new RDMPageVO( curPage: 1, Integer.MAX_VALUE);
    return exaDefinitionLinkDelegator.find(queryRequestVo, rdmPageVO);
}
```

```
public ArrayList<EXADefinitionViewDTO> getClassificationAttribute(String classificationID) {
    List<EXADefinitionLinkViewDTO> v = attributeMapper.getAttributeLinkByTargetID(classificationID);
    ArrayList<EXADefinitionViewDTO> ret = new ArrayList<>();
    if (v == null) {
        return ret;
    }
    for (EXADefinitionLinkViewDTO d : v) {
        ret.add(d.getSource());
    }
    return ret;
}
```

7. 文件服务相关代码参考

（1）上传文件：

```
public RDMResultVO uploadFile(CustomFile customFile){
    MultipartFile file=customFile.getFile();
    UploadFileModelVO uploadFileModelVO=new UploadFileModelVO();
    uploadFileModelVO.setFile(file);
    uploadFileModelVO.setModelNumber("DM02192458");
    uploadFileModelVO.setModelName("DesignBlueprint");
    uploadFileModelVO.setAttributeName("BluePrint");
    uploadFileModelVO.setApplicationId("xxx");
    uploadFileModelVO.setUsername("xxx");
    uploadFileModelVO.setStorageType(0);
    uploadFileModelVO.setExaAttr("0");
    uploadFileModelVO.setInstanceId(customFile.getId());
    uploadFileModelVO.setEncrypted(false);
    MultipartFile[] multipartFiles =new MultipartFile[1];
    multipartFiles[0]=file;
    uploadFileModelVO.setFiles(multipartFiles);
    System.out.print(uploadFileModelVO);
    RDMResultVO a=fileDelegatorService.uploadFile(uploadFileModelVO);
    return a;
}
```

（2）下载文件：

传输文件的时候建议使用自定义类传输文件和对应的实体 id

```
新 *
@PostMapping(@~"/downloadfile")
public void downloadFile(@RequestParam("file_ids") String fileids, @RequestParam("instance_id") String id, HttpServletResponse response) {
    try {
        // 假设 fileDelegatorService.downloadFile 会正确设置响应头和写入数据
        fileDelegatorService.downloadFile(fileids, modelName: "DesignBlueprint", attributeName: "BluePrint", id, applicationId: "50dddc8491764@b880e853db343f42d", isMasterAttr: "0", response);
    } catch (Exception e) {
        // 处理异常，例如设置错误状态码和错误消息
        response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        try (PrintWriter out = response.getWriter()) {
            out.write( s: "Error occurred while downloading file: " + e.getMessage());
        } catch (IOException ioEx) {
            // 记录或处理 IOException
            ioEx.printStackTrace();
        }
    }
    // 注意：不需要返回 response，因为响应已经被写入
}
}
```

六、关于扩展及加分功能

（1）除上述简单功能外，iDME 还有更多功能供各位同学探索，例如版本管理和检入检出，数据模型的生命周期管理，使用 BomLink 制作 Bom 树等等。

（2）各位同学可通过以下链接和相关开发文档（附件）进行查看，根据情况选择更多功能进行探索尝试和实现。链接（可搜索所有功能）：[生命周期管理开发指导_工业数字模型驱动引擎_华为云]